**Master-slave architecture:**

**Adding slave Linux node:**

Prerequisite:

1. Create one new node..

In order to use your node as a Jenkins slave, you will need to install the following initial dependencies:

Install Java and git.

| sudo dnf install java-11-openjdk-devel -y sudo dnf install git-all -y |
| --- |

Note down ip of master and nodes:

master :

| For ex: ssh -i "Jenkins.pem" ec2-user@ec2-52-15-50-45.us-east-2.compute.amazonaws.com |
| --- |

Slave:

For ex:

| ssh -i "Jenkins.pem" ec2-user@ec2-3-143-215-201.us-east-2.compute.amazonaws.com |
| --- |

1. Login to master and create ssh-keygen

| ssh-keygen -t rsa -b 4096 -C "<add-your-node-name>" |
| --- |

My example: ip-172-31-18-87

[ec2-user@ip-172-31-18-87 ~]$ hostname

Ip-172-31-18-87.us-east-2.compute.internal

1. Copy Masters public key.

| [ec2-user@ip-172-31-18-87 ~]$ cat .ssh/id\_rsa.pub  ssh-rsa  ip-172-31-18-87 |
| --- |

1. Add this key to the slave’s authorised keys

Login to slave

| [ec2-user@ip-172-31-27-11 ~]$ cat .ssh/authorized\_keys  ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCmDkBtu4QyA+2O2hrUEOQ3aNoeA7pPcnZvita3uVH5zNvghL7IQUE7d+kfLujehAZY+LRKnxBhVMd77V3td5hfu5OgDIa2V490gMKxD1sqLfLKbrEF6Vb8pMV7YoGwX6925/cOHTrz3pyi8ow3npFYYlAShj9Pw0x8G/3aRQfOdTpDEEAzOtdKvlBh4qKLRxTBAb/S7V3ypvBBFmSzhlcl/IEiCmj1QGCxkkjIVxj+qKI+gkXm1YWy3KvHmvz89g+2J+/wSJkzqm+U7qx0BrrLPuEfROkK0efWBeF+Z/74cd9TlMUDPDi0NdLQ87LrxrUZKfwKNlPZPQnay/0f4Gk/ Jenkins ssh-rsa  ip-172-31-18-87 |
| --- |

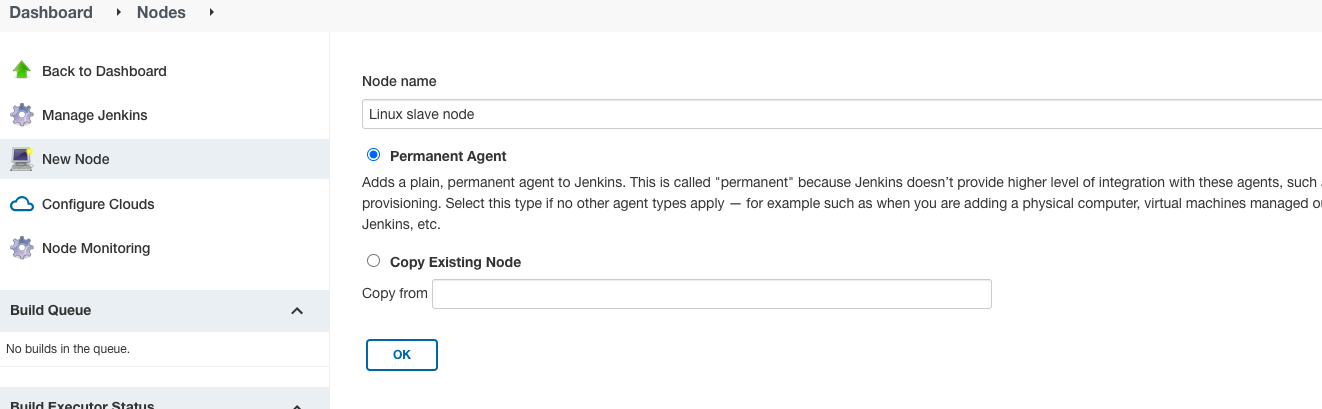
Crete a new directory on slave.

| [ec2-user@ip-172-31-27-11 ~]$ mkdir /home/ec2-user/jenkins/ |
| --- |

Login back to the master: and copy private key.

| [ec2-user@ip-172-31-18-87 ~]$ cat .ssh/id\_rsa -----BEGIN OPENSSH PRIVATE KEY----- b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAACFwAAAAdzc2gtcn NhAAAAAwEAAQAAAgEAoR1vVJZaE5DT63bRxlNyDGj95EHVxhYTsavFbf8arprgy/OnDRAC VNMZFPEHqHVyXKt2WYpY+1d/4SXjwBkdLUVAaozgoAlbf2uoGm1yylXcx8KfyEe6jrMT+s ry0VuFZ+CdNOG+LUH3U9pE1NX3AgFr7ujeJeGPMon6vUNo4g0lv2zAcb0C44AIeqD6ajxQ sj1q5j8qjaLuHGDAjGg++cRXHZ8M96v2HlEfGFbSXyF4wGfLmaR35vCdJh1CQCA3qWOGyn IQyzRmeD6WwNhGv1OAYrhpkiCMao0lSDVI7f1syechR0cNAoFkPqjtczh+C5IGB8KhDq3m b3BIW1+NZQCH6WNPhMrcyuvT7G4hvww09dz6PNBgFoWVD894cFXnmxkmde1oYnu72/nEyW JrAKPZnBBLhjnHwjDe2Zm2YIfq++MRhfo9VasknajZRK84ZsByrjYvA7WRNeLBd7SRCTzp K+U+ZYVN61BHibuLpyvtyKoAbut8hg5iPcsysCnTgvviqUpchChG57K6cjeC9BU3JoHavF MXBKskh+zJNT8aBnHH8KjE5yodtatoLHUfn4crTnpz9Xoq25ge8bdYJHrBighrmNAbKtzu  —-----truncated output—------ |
| --- |

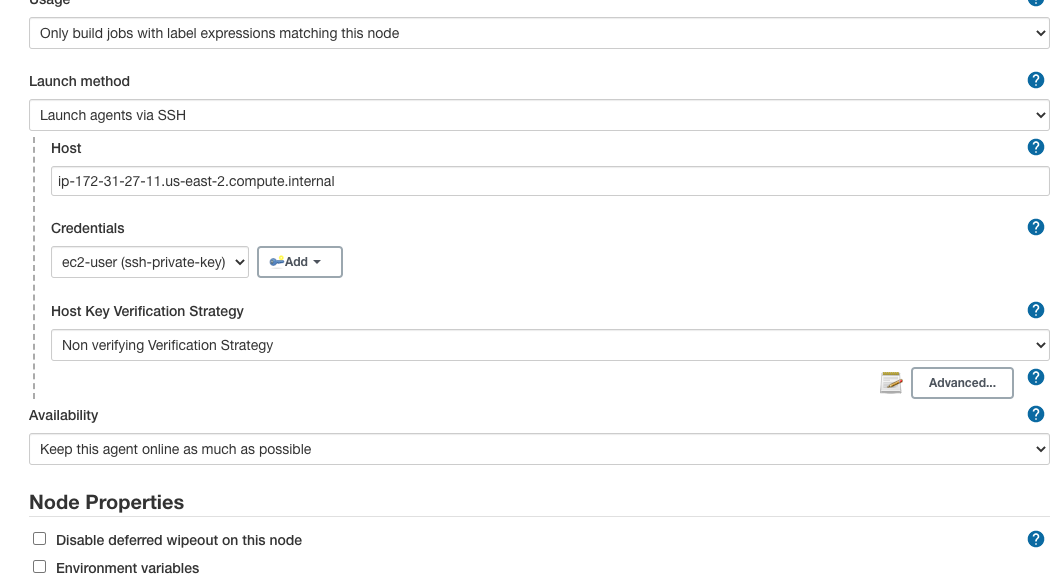
Create new node in jenkins:



Add following configuration:

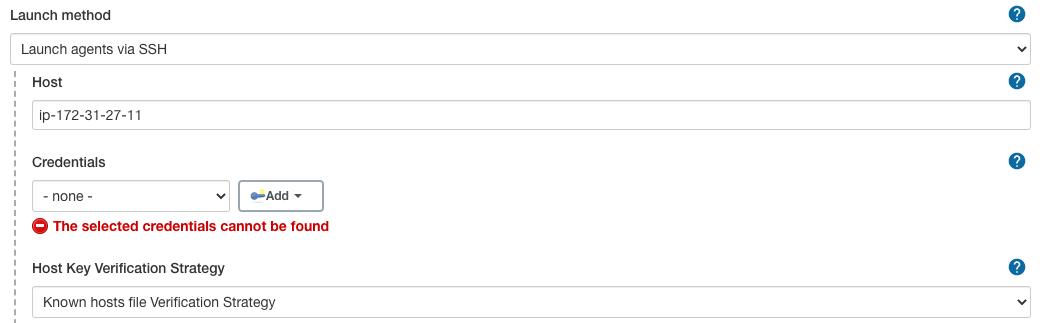


Name Can be anything use: slaveNode1.

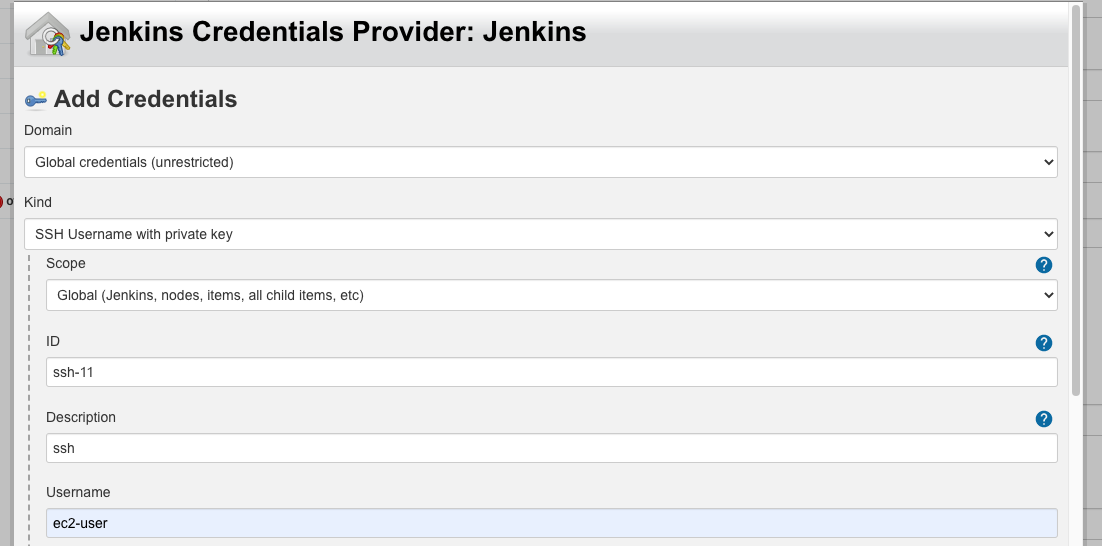


Launch Method:

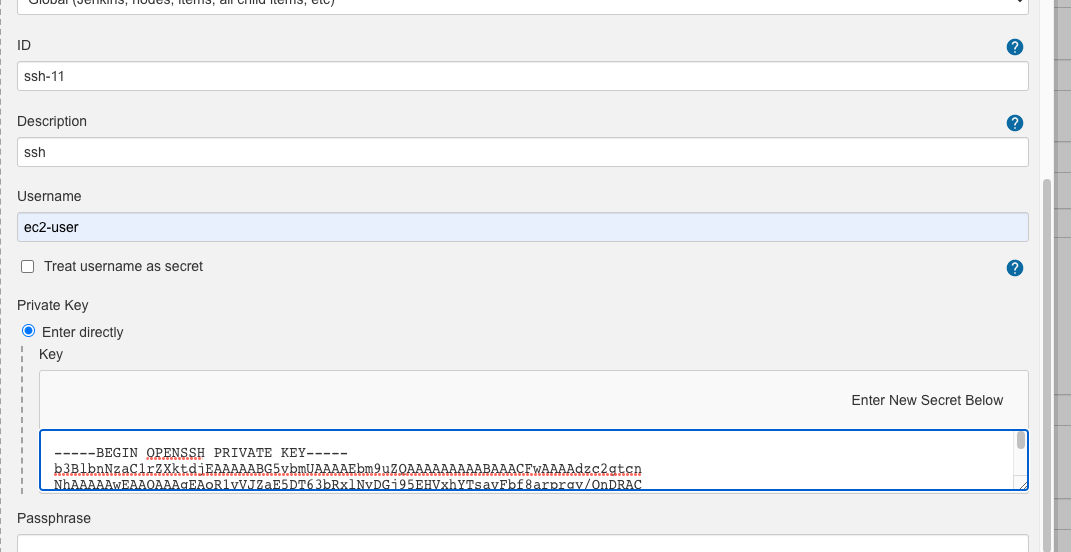
Launch agent using ssh:



Add credentials:



Add private-key



Save.

Node should be online in some time.

Running pipeline on Slave node:

## **Building the Project on Slave Nodes**

Now that our master and slave nodes are ready, we'll discuss the steps for building the project on the slave node.

For this, we start by clicking “New Item” in the top left corner of the dashboard.

Next, we need to enter the name of our project in the “Enter an item name” field and select the “Pipeline project”, and then click the “OK” button.

On the next screen, we'll enter a “Description” (optional) and navigate to the “Pipeline” section. Make sure the “Definition” field has the Pipeline script option selected.

After this, we copy and paste the following declarative Pipeline script into a “script” field:

node('slaveNode1'){

stage('Build') {

sh '''echo build steps'''

}

stage('Test') {

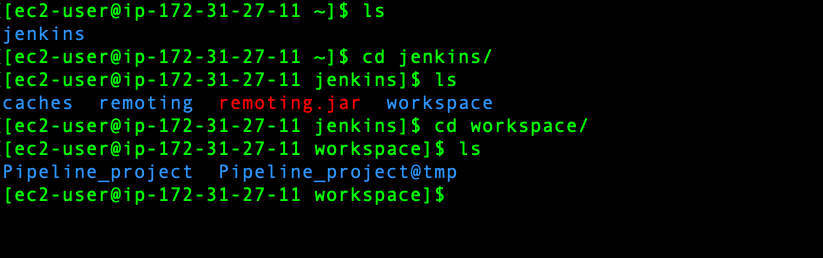
sh '''echo test steps'''

}

}

Next, we click on the “Save” button. This will redirect to the Pipeline view page.

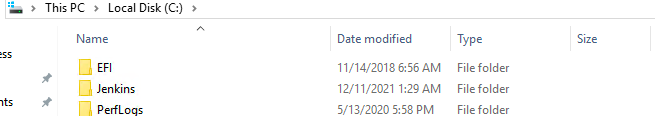
Save and run the pipeline. You can view the files created in linux slave node.



**Configuring windows slave:**

Prerequisite on slave:

1. Create one Jenkins directory.

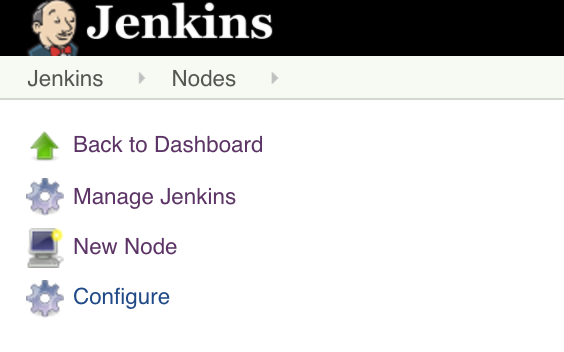


**On Jenkins UI:**

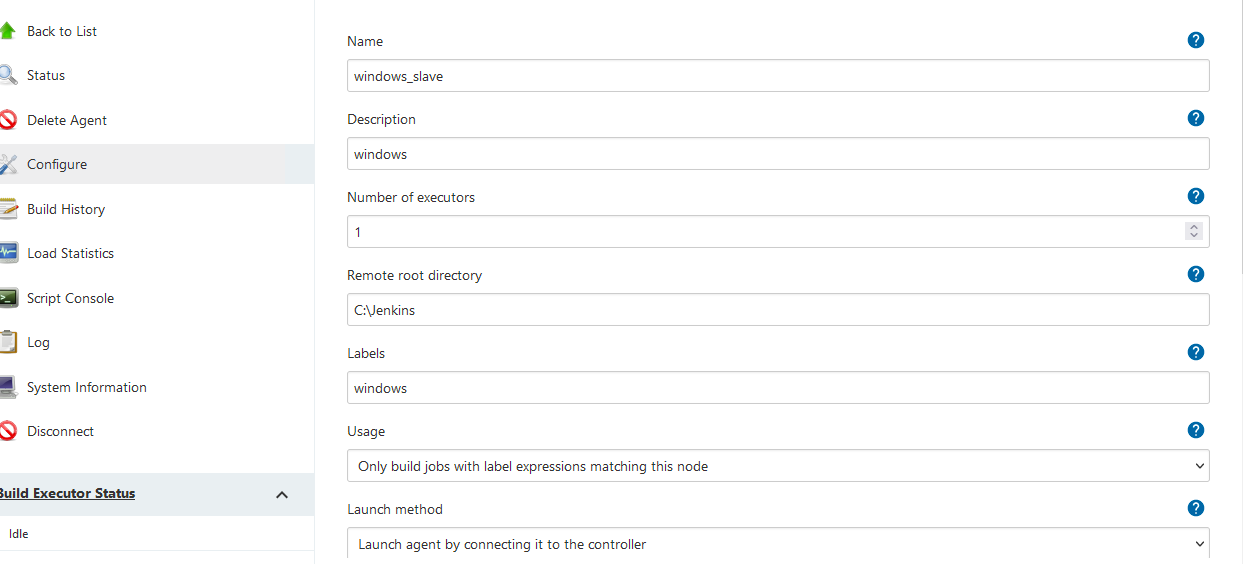
1- Login to Jenkins and Click on Manage Jenkins

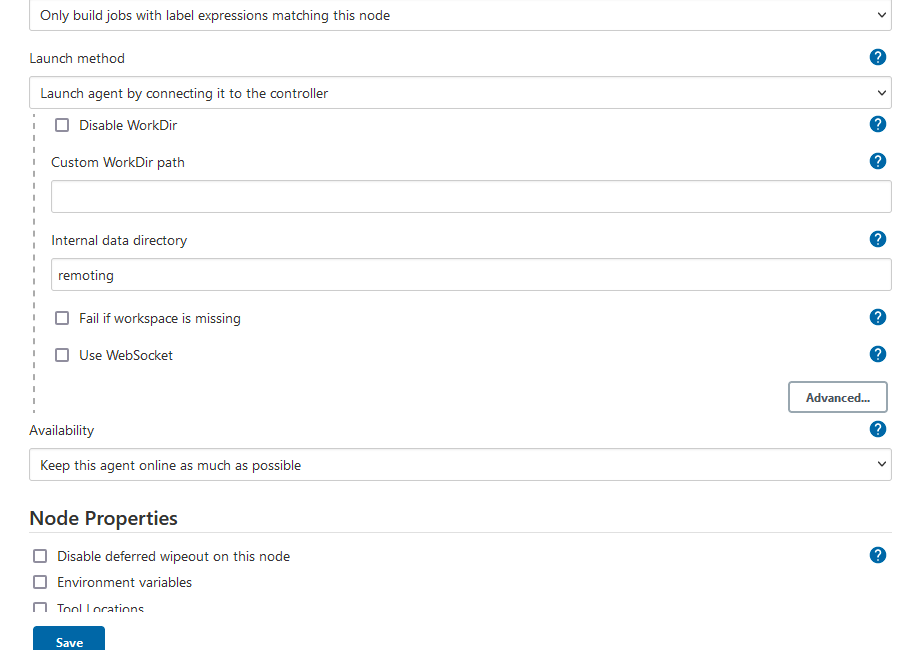
jenkins-windows-slave-creation-1

2- To create a new Slave – Click on New Node



4. Name the node:



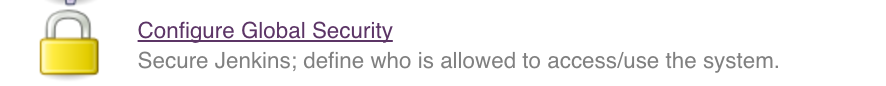
l

Save the configuration.

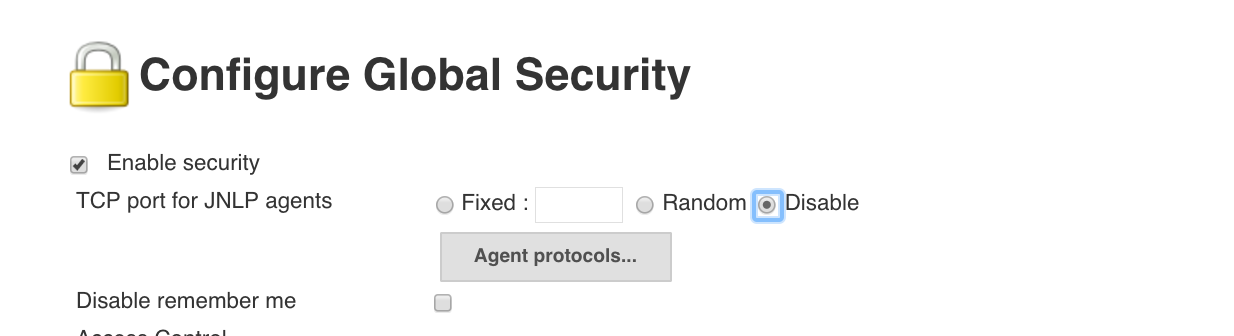
5 – Again click on **Manage Jenkins**

**jenkins-windows-slave-creation-1**

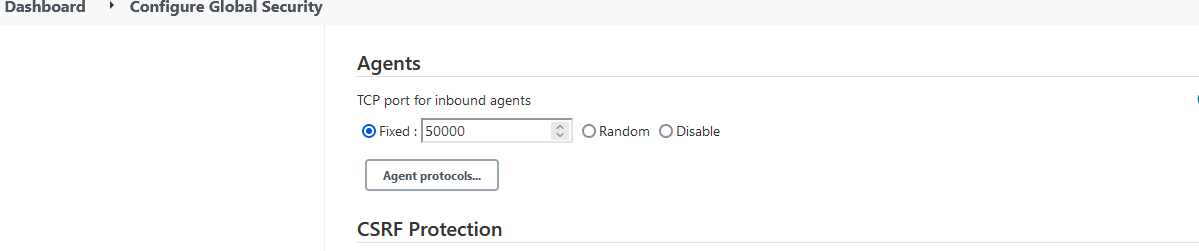
6- Then click on **Configure Global Security**

****

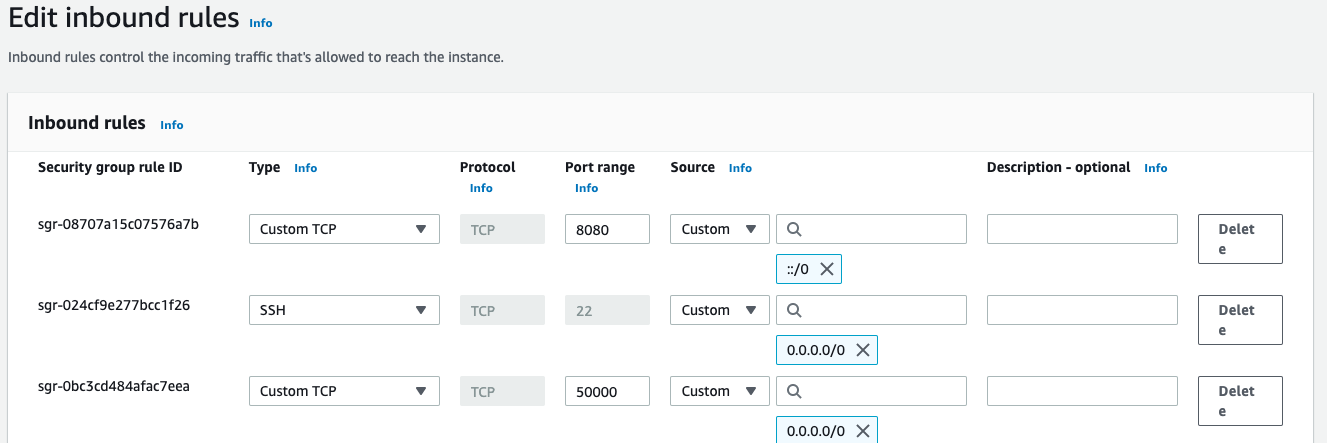
7- Find out the **Enable Security** option



8- And then change the



9. Open 50000 port in security group of master.



10- Once the new node is created, it will be in **Offline** mode

11. Login into the slave windows node:

12. Login into jenkins and download the jar file from the failed node add section.

13. Use the command mentioned in the failure message to connect to the master.

14. Check if the master slave configuration is working.

**Exploring more on Pipeline - Jenkinsfile.pdf and syntax\_compare.pdf**

**Demo 1: Scheduled build:**

**Installing maven:**

~~~~~~~~~~~  
$ wget https://download.java.net/java/GA/jdk13.0.1/cec27d702aa74d5a8630c65ae61e4305/9/GPL/openjdk-13.0.1\_linux-x64\_bin.tar.gz

$ tar -xvf openjdk-13.0.1\_linux-x64\_bin.tar.gz

$ mv jdk-13.0.1 /opt/

~~~~~~~~~~~~~

* Open .profile file from the home directory and add the following lines to it.

~~~~~~~~~~~~~~~~

* JAVA\_HOME='/opt/jdk-13.0.1'
* PATH="$JAVA\_HOME/bin:$PATH"
* export PATH

~~~~~~~~~~~~~~~~

$ java -version

* Install Maven:

~~~~~~~~~~~~~~~  
$ wget https://mirrors.estointernet.in/apache/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz

$ tar -xvf apache-maven-3.6.3-bin.tar.gz

$ mv apache-maven-3.6.3 /opt/

~~~~~~~~~~~~~~~

~~~~~~~~~~

M2\_HOME='/opt/apache-maven-3.6.3'

PATH="$M2\_HOME/bin:$PATH"

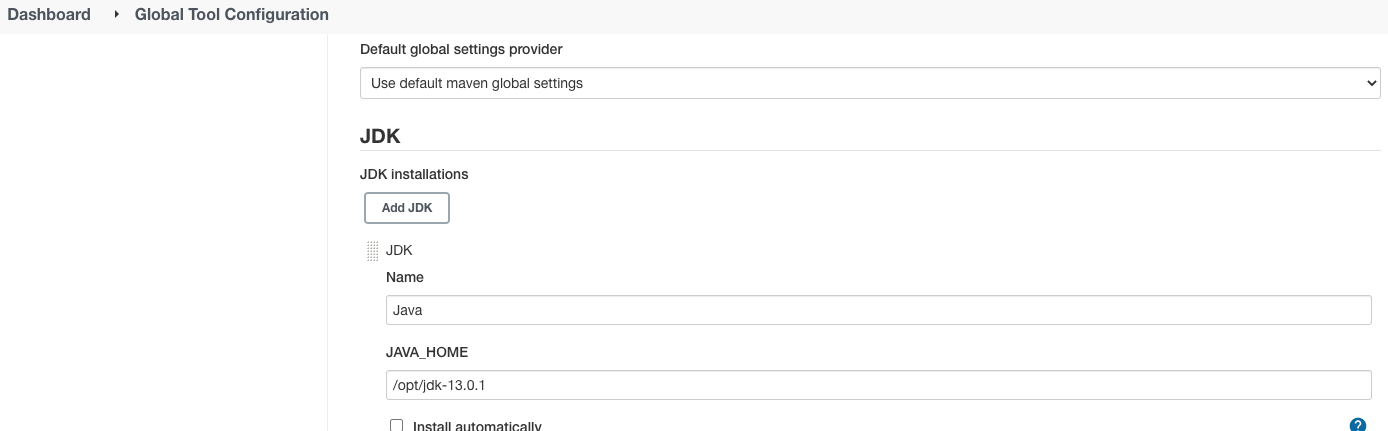
export PATH

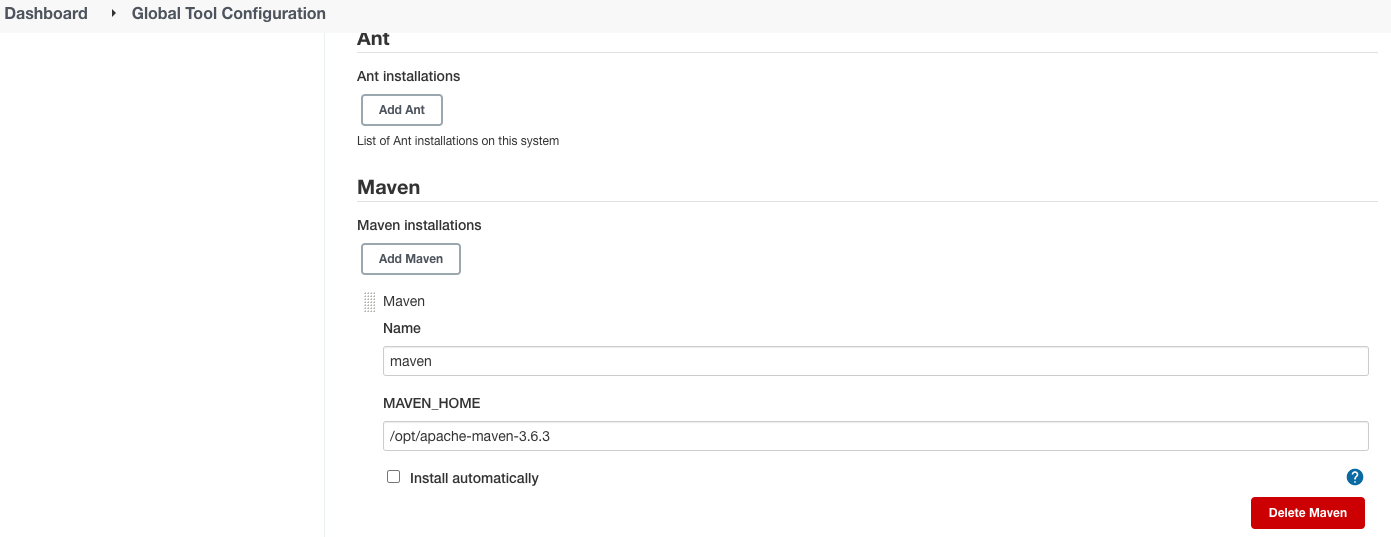
~~~~~~~~~

$ mvn -version

**Step 1:** *Configure Jenkins*

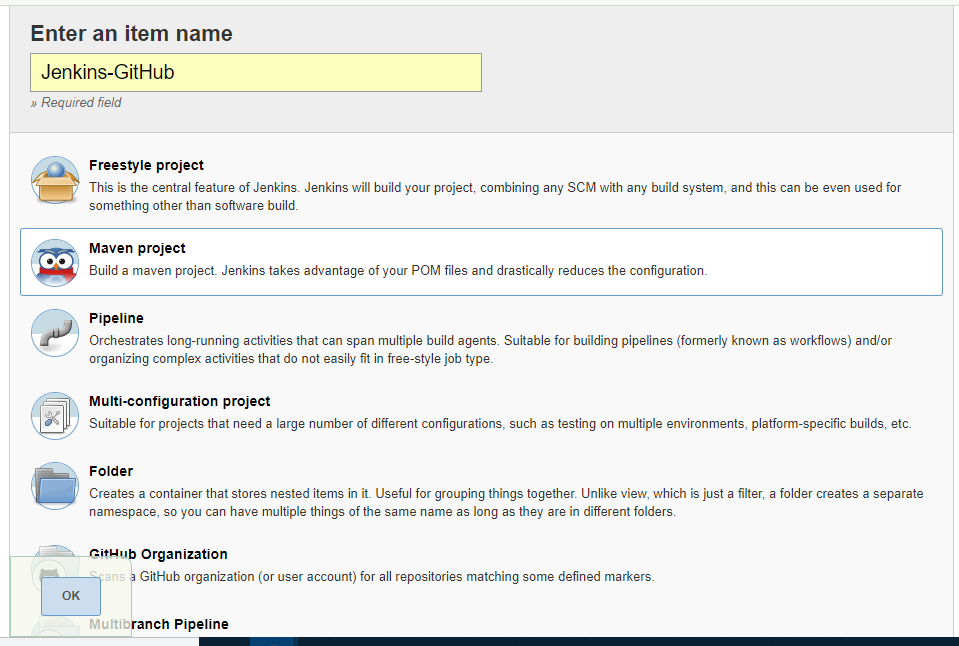
1. Insure that **GitHub Plugin** is installed under- **Manage Jenkins** > **Manage Plugins** > **Installed** search for git. If not installed move to **Available** Tab and search for git and install it.
2. Configure *Java*, *GitHub* and *Maven* for Jenkins
3. Navigate to **Manage Jenkins** > **Global Tool Configuration** > Under **JDK**section provide *Name* and path to *JAVA\_HOME*, in same way for Git provide *Git Name* and path to Git executable, same in case of **Maven** provide *Name*and *MAVEN\_HOME* as in below images



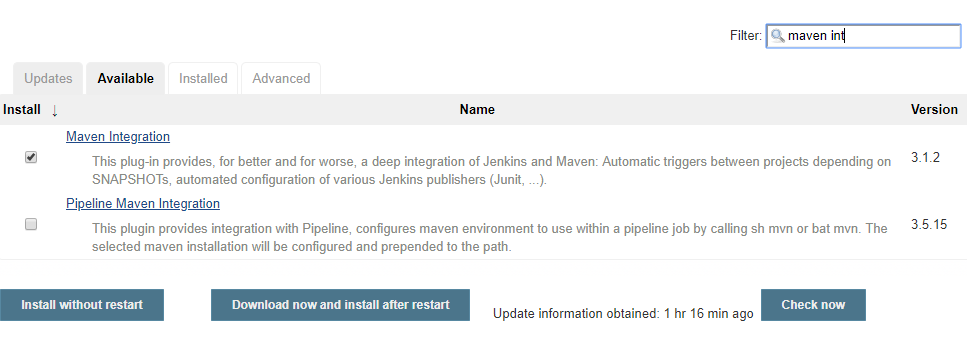


**Step 2:** *Create Job*

1. Create a new Job by clicking **New Item**
2. Enter your Job Name **‘Jenkins-GitHub’** and select ***Maven Project*** then click **OK**, You will be navigated to configure the Job.



If *Maven Project* is **not visible** in your Jenkins then install plugin named as ***‘Maven Integration’***

******

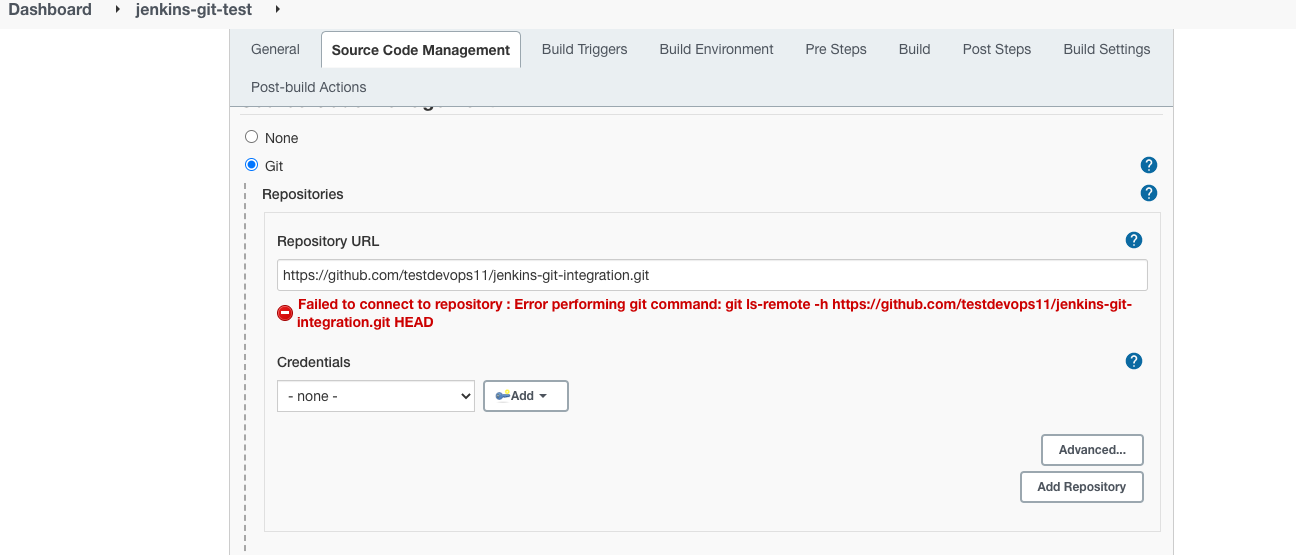
3. In job configuration under **General** Section tick **Github** project and provide your *project url* from Github- [***https://github.com/testdevops11/jenkins-git-integration***](https://github.com/testdevops11/jenkins-git-integration)

this will provide you link to GitHub from Job dashboard and it is optional

4. Under **Source Code Management** section click on **Git** radio button and provide Repository URL- <https://github.com/testdevops11/jenkins-git-integration>

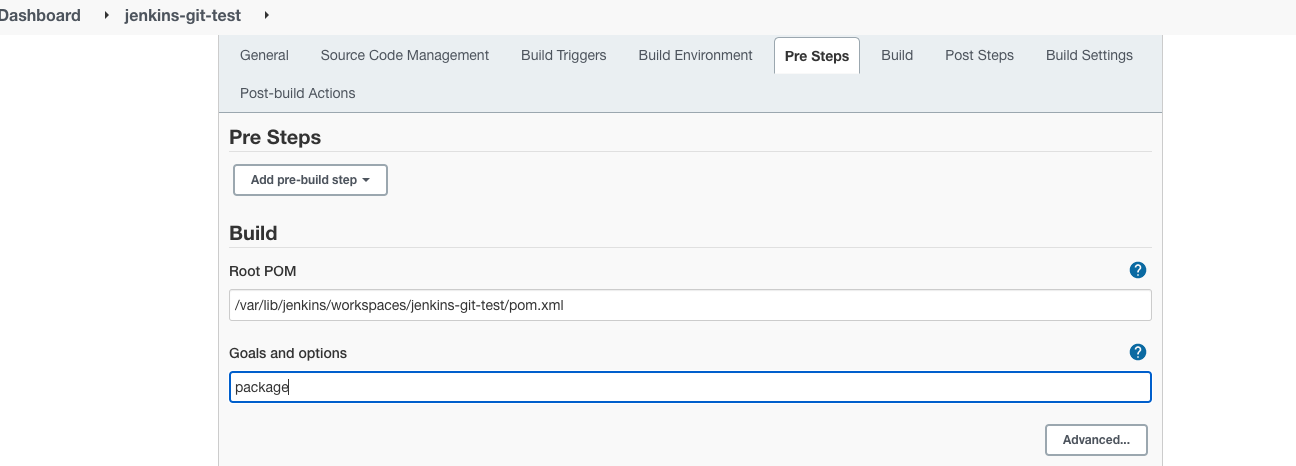
5. Select *branch* to build if master- ***‘\*/master’*** if development- ***‘\*/development’***

6. You can add *behavior* by selecting value from drop down what to perform in my case I want to clean space before code checkout so I have selected ***Clean before chekout*** (you can configure according to your need this is optional)



7. Navigate to **Build Trigger**- If you want to build your project on *specific time interval* you can configure it under **Build Trigger** > tick **Poll SCM** and in text field ***\* \* \* \* \**** (five star separated by space for every minute, see bottom of this article for convention). If you will not set up Poll you will have to manually build the Job.

8. Navigate to **Build** Section > provide path to ***pom.xml*** (for maven project) and *Goals and Options* **(**you can provide relative or absolute path to pom.xml, ***Relative*** : from project checkout directory as in below image, **Absolute** : from base directory like ***/var/lib/jenkins/workspace*** and for goals you can define as you have set in your ***pom.xml***like **clean**, **package**, etc**)**

****

Navigate to that project. You will have to click on **Build Now** if you have not configured *Poll SCM* else it will automatically trigger every minute (as we have set \* \* \* \* \*). Your project will build successfully.

Go to project path /var/lib/jenkins***/workspace/Jenkins-GitHub/target***) and under target you will find the ***jar*** or ***war*** of your build project.